Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 JUN 2006	**N/A**	**-**

4. TITLE AND SUBTITLE
Electrical Modeling and Simulation With Matlab/Simulink and Graphical User Interface Software

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Ueda,Jason; Daniszewski,David; Monroe,John; Masrur,Abul; Charbeneau,Eric; Jochum,Eric; Patel,Rakesh

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army RDECOM-TARDEC 6501 E 11 Mile Rd Warren, MI 48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
15987

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)
TACOM/TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
15987

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**3**	

Paper Offer #: 06PSC-51

Electrical Modeling and Simulation With Matlab/Simulink and Graphical User Interface Software

Jason Ueda, David Daniszewski, John Monroe, Abul Masrur, Michelle Charbeneau, Eric Jochum, Rakesh Patel
US Army TARDEC

ABSTRACT

This paper describes modeling and simulation technologies used to simulate the electrical systems of Army vehicles using Matlab/Simulink coupled with graphical user interface software. The models were built using Mathworks' Matlab/Simulink software in conjunction with the SimPowerSystems Toolbox, a toolkit provided by Mathworks that provides models of basic electrical components such as capacitors and inductors, in addition to more advanced components such as diodes and IGBT's. The current results of this ongoing effort are presented and discussed.

INTRODUCTION

Fast and accurate modeling of electrical systems can prove complicated, due to the linear and non-linear elements in the system. For instance, suppose a parallel RLC circuit is represented by the following equation:

$$\frac{V_c}{R_s} + \frac{1}{L}\int V_c dt + C\frac{dV_c}{dt} = \frac{V_s}{R_s}u(t-0.1), \text{ [1]}$$

Modeled in Simulink, it might resemble Figure 1.

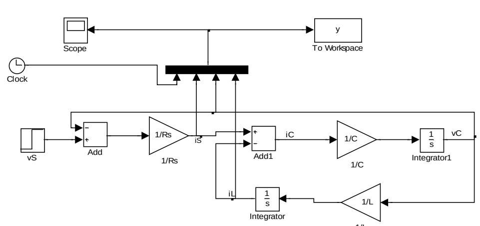

Figure 1

If the system were comprised only of one parallel RLC circuit, then this would simplify matters. However, vehicular electrical systems are more complicated and may have multiple series and/or parallel RLC circuits, in addition to electronic devices that are more accurately described with non-linear equations. To directly enter the differential equations corresponding to vehicular electrical systems in Simulink would be time-consuming and probably cumbersome, not to mention difficult to debug for particularly complicated system schematics. Use of SimPower allows modeling the physical electrical system directly and the software constructs the appropriate equations.

SIMPOWER

This toolkit provides "black box" components such as voltage sources, resistors, inductors, diodes, etc. which enable a user to construct an electrical model by simply connecting the proper simulated components together, mimicking an electrical schematic.

Similar to the concept of "black box" or object-oriented methodology in software-engineering, SimPowerSystems lets the user "plug" the electrical components together, to make a functioning model, while the state-space equations which model the Simulink circuit are automatically generated and remain hidden from the user. This enables a much more rapid development of an electrical model from a schematic.

COMPONENT MODELING

During the course of the effort, challenges were encountered with respect to certain devices not provided by the SimPowerSystems Toolbox. Common power electronic components such as rectifiers and inverters were developed to ensure a smooth system integration effort. These components were developed to the appropriate level of fidelity to ensure that although the model produced accurate results, it did so in a reasonable time frame.

INVERTER

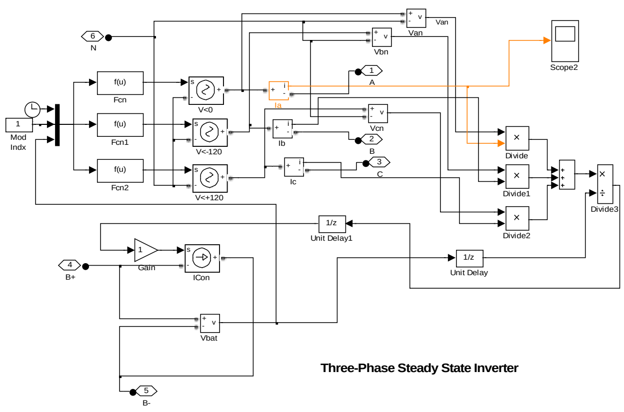

Figure 2 Three-Phase Steady State Inverter

Typically three-phase inverters are used to supply three-phase loads. For vehicles, the inverter is typically needed to convert DC power to AC power for the typical induction motors used in hybrid electric applications.

During the course of the investigation, a three-phase inverter model was developed as a component level-drop in for various vehicle models in development consideration. Due to some overhead introduced by SimPowerSystems, it was decided to use Simulink to develop a mild three-phase inverter, and in this instance the appropriate equations were known. The following equations describe the duty cycles of the three phases, where Vbat is the voltage of a vehicle battery and t is time:

$$0.5 * \mathrm{Vbat} * (\sin(2 * \mathrm{pi} * 60 * \mathrm{t}))$$

$$0.5 * \mathrm{Vbat} * (\sin(2 * \mathrm{pi} * 60 * \mathrm{t} - 2 * \mathrm{pi}/3)), \quad [2]$$

$$0.5 * \mathrm{Vbat} * (\sin(2 * \mathrm{pi} * 60 * \mathrm{t} + 2 * \mathrm{pi}/3))$$

STATEFLOW

In augmenting the relevance of these models, test cases using various hypothetical mission scenarios had to be studied. The development of these scenarios and feeding it into the models via a usable data interface was another challenge in itself. One application tool studied was the use of Stateflow, a toolbox provided by Mathworks which aids with the type of event-driven modeling a mission scenario requires.

Stateflow is a graphical design tool enabling the user to model event-driven behavior. It is based on finite state machine (or finite automaton), whereby each “state” reflects the condition a system is in. A state may be entered or exited when certain (non-)conditional events occur which require a change in the system.

For the purposes of the ongoing effort, Stateflow is being used as a control signal interface to the SimPowerSystems models. At a certain time t, a vehicular load will be turned on/off, and Stateflow sends the appropriate command signal to the model.

In an attempt to enable a user to have a friendly interface in order to evaluate the model results, it was decided that being able to read a mission scenario from an Excel spreadsheet would be a feasible solution.

EVENT MODELING

The decision to look at spreadsheet based input forced a number of programming issues in terms of setting up a simulation design via Stateflow and Simulink. First, a programming solution to extract data from the spreadsheet and feed it to the Simulink model had to be created. Next, due to programming language constraints, a specific format for the data had to be created. Finally, a parallel state chart had to be set up to send overlapping signals to the Simulink model (the term “overlapping” in this case refers to the fact that signals indicated certain loads were turned on/off simultaneously or at overlapping time intervals). Within the state chart, dynamically-sized arrays were created depending upon the number of on/off events there were for a particular device.

API Programming

Device #	Name	Start	Duration	On/Off
1	Master Power	00:00:10	00:00:20	1
1	Master Power	00:00:30	00:00:10	0
2	Fuel Pump	00:00:10	00:00:20	0
2	Fuel Pump	00:00:30	00:00:10	1

Figure 3 Spreadsheet Input

It was decided that vehicle loads that would be turned on/off would have a start time and a duration time (how long it was supposed to be on/off). As mentioned previously, the on/off events for the devices could overlap, which would be realistic for a given mission scenario. Figure 3 shows a simple test case developed to test the interaction of the Matlab programming scripts, Stateflow toolbox, and Simulink model.

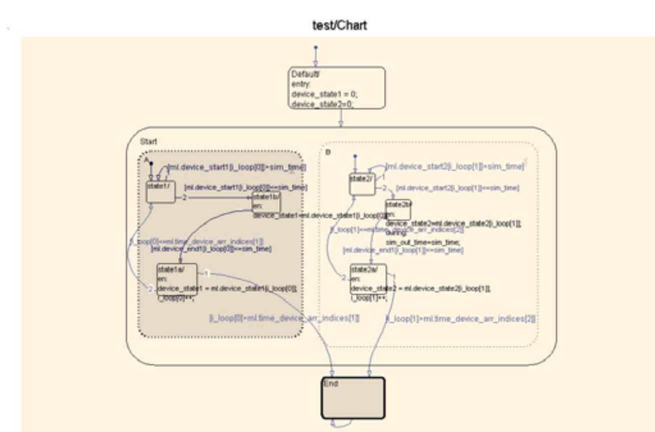

Figure 4 Stateflow Parallel State Chart

Since it was not known in advance how many times a particular device load would be turned on and off, it was necessary to use the Stateflow API (application programming interface) in order to dynamically size array variables which tracked the on and off times of the loads. The API enables a programmer to access a Stateflow chart via Matlab scripting commands which sift through the Stateflow objects.

Path forward

Some of the challenges associated with the Stateflow application included computation of parallel states, which led to overlapping signals being sent after their appointed time. This timing issue appears to be a function of the way Stateflow evaluates parallel states and may simply be a hard limitation which cannot be overcome. In addition, a methodology for describing non-discrete events may have to be conceived (e.g., such as for a light dimmer switch where there is a range of lighting available between on and off). The Stateflow API is a powerful tool that can be used to dynamically create a state chart on the fly, and the possibility of creating a state chart from a script (as opposed to the manual creation employed in this effort) exists.

LABVIEW

In attempting to speed up the development of a graphical user interface, it was decided to study Labview. Labview is a software tool typically used in data acquisition and analysis environments and enables a visual GUI to be developed rapidly via its plug and play components and gauges. Thus, to demonstrate the performance of the models to the casual user, a graphical user interface was developed that displayed the relevant system data. However, integration issues with Matlab/Simulink presented additional obstacles that had to be overcome with mapping conventions using National Instruments' Simulation Interface Toolkit, the software interface designed to link Labview with Matlab/Simulink. For instance, it was found that occasionally trying to map a gauge readout to a Simulink / SimPowerSystems measurement component caused inconsistent system errors.

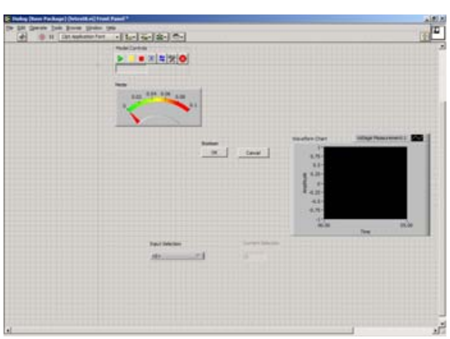

Figure 5 Example Labview Screenshot

In addition, the capabilities of the Java and Python programming languages are still being explored as alternatives to Labview.

CONCLUSION

Constructing an integrated event-driven model with multiple software toolkits presented interesting challenges from a systems engineering perspective. Tools which were thought to work right out of the box with Simulink (such as the Simulation Interface Toolkit), required more finesse and debugging than anticipated. Though developing a baseline model seems readily accessible to the casual user, a more thorough model which may one day be used to augment design trade studies may require a slightly altered approach.

For instance, perhaps Labview does not scale well when required to interface with hundreds of vehicle loads in the model. This may force a user to simply using just the native graphics capabilities with Matlab in order to prevent this. Future work should include a more integrated and detailed system.

REFERENCES

1. Stateflow and Stateflow Coder API. Version 6. March 2006. The Mathworks, Inc.
2. Stateflow and Stateflow Coder User's Guide. Version 6. March 2006. The Mathworks, Inc.
3. Stateflow User's Guide. Version 6. March 2006. The Mathworks, Inc.
4. Scott D. Sudhoff, Dionysios C. Aliprantis, Brian T. Kuhn, and Patrick L. Chapman, "An Induction Machine Model for Predicting Inverter-Machine Interaction," *IEEE Transactions on Energy Conversion*, vol. 17, no.2, pp. 203-210, June 2002.
5. Heath Hofmann, Richard Stroman, and Michael Lanagan, "Closed-Form Frequency Model of 3-Phase Inverter Drive for DC Distribution System Analysis," Paper 2002-01-3232, SAE Power Systems Conference Oct. 29-31, 2002. Reprinted from Power Systems Conference Proceedings on CD-ROM (PS2002CD).